تحلیل باج‌افزار ویندوزی Speedy

تاریخ گزارش: ۳۰ اردیبهشت ۱۳۹۹

مقدمه

به تازگی آزمایشگاه امنیت کی‌پاد بدافزاری برای پلتفرم ویندوز در زیرساخت سطح کاربران این سیستم‌عامل در کشور جمهوری اسلامی ایران رصد و شناسایی کرده است که این بدافزار قابلیت‌های ویژه و منحصربفردی دارد. شایان ذکر است، در هنگام بررسی اولیه که توسط آزمایشگاه امنیت کی‌پاد صورت گرفت، این بدافزار توسط ضدباج‌افزار کی‌پاد به صورت کامل قابل شناسایی بوده است.

شایان ذکر است، باج‌افزارهای مشابه با این باج‌افزار با نام‌هایی از قبیل STOP، DJVU، Drume و StopData تاکنون مشاهده شده است که تقریبا یک ساختار و عملکرد مشابه با یکدیگر دارند و فرمت فایل opqz، mpal، mogranos، cosakos، nvetud، lotej، kovasoh، prandel، zatrov، masok، brusaf، londec، یا krusop تنها بخشی از فرمت فایل‌هایی هستند که این بدافزار استفاده می‌کند.

شایان ذکر است، این باج افزار مانند انواع دیگر از باج‌افزارهای خانواده STOP/DJVU از طریق هرزنامه‌ها و فیشینگ گسترش پیدا می‌کند. پس از آلوده شدن سیستم قربانی، این باج‌افزار از الگوریتم‌های بر پایه AES-256 و RSA یا یک الگوریتم جدیدتر برای رمزگذاری فایل‌های میزبان استفاده می‌کند. سرعت عملکرد این باج‌افزار آنقدر بالاست که به محض آلوده شدن سیستم، قربانی می‌تواند تغییر پسوند فایل‌هایش به .mpal یا .opqz و دیگر پسوندها را مشاهده کند. در ادامه باج افزار فایل‌های پشتیبانی سایه (Shadow) قربانی را نیز حذف می‌کند تا احتمال بازگردانی فایل‌ها غیرممکن شود.

هنوز آماری در رابطه با تعداد قربانیان این باج‌افزار در ایران منتشر نشده است اما قربانیان می‌گویند هکرها مبالغی بین ۳۵۰ تا ۸۰۰ دلار را از آن‌ها به عنوان باج درخواست کرده‌اند. شروع گسترده آلودگی سیستم‌ها به باج‌افزارهای این گروه از سال ۲۰۱۸ بود و از این رو گروه تصمیم گرفت نسخه‌های مختلف و پیچیده‌تر آن را با پسوندهای مختلف و البته الگوریتم‌های پیچیده‌تر منتشر کند.

به هر صورت، در حال حاضر باج‌افزارها تهدید جدی به شمار می‌روند، و هر از گاهی قربانیان زیادی از کسب‌وکارهای کوچک تا شرکت‌های تجاری بسیار بزرگ می‌گیرند. باج‌افزاری که در ادامه به تحلیل آن خواهیم پرداخت تاکنون توانسته است برخی از سازمان‌های رسمی کشور را مورد هدف قرار بدهد و با موفقیت آن‌ها را آلوده کند.

مشخصات جاسوس‌افزار

در جدول زیر، مشخصات کلی باج‌افزار ویندوزی Speedy به صورت خلاصه آورده شده است. در ادامه، تحلیل این بدافزار که از خانواده باج‌افزارها به شمار می‌رود، با جزئیات دقیق‌تری آورده شده است.

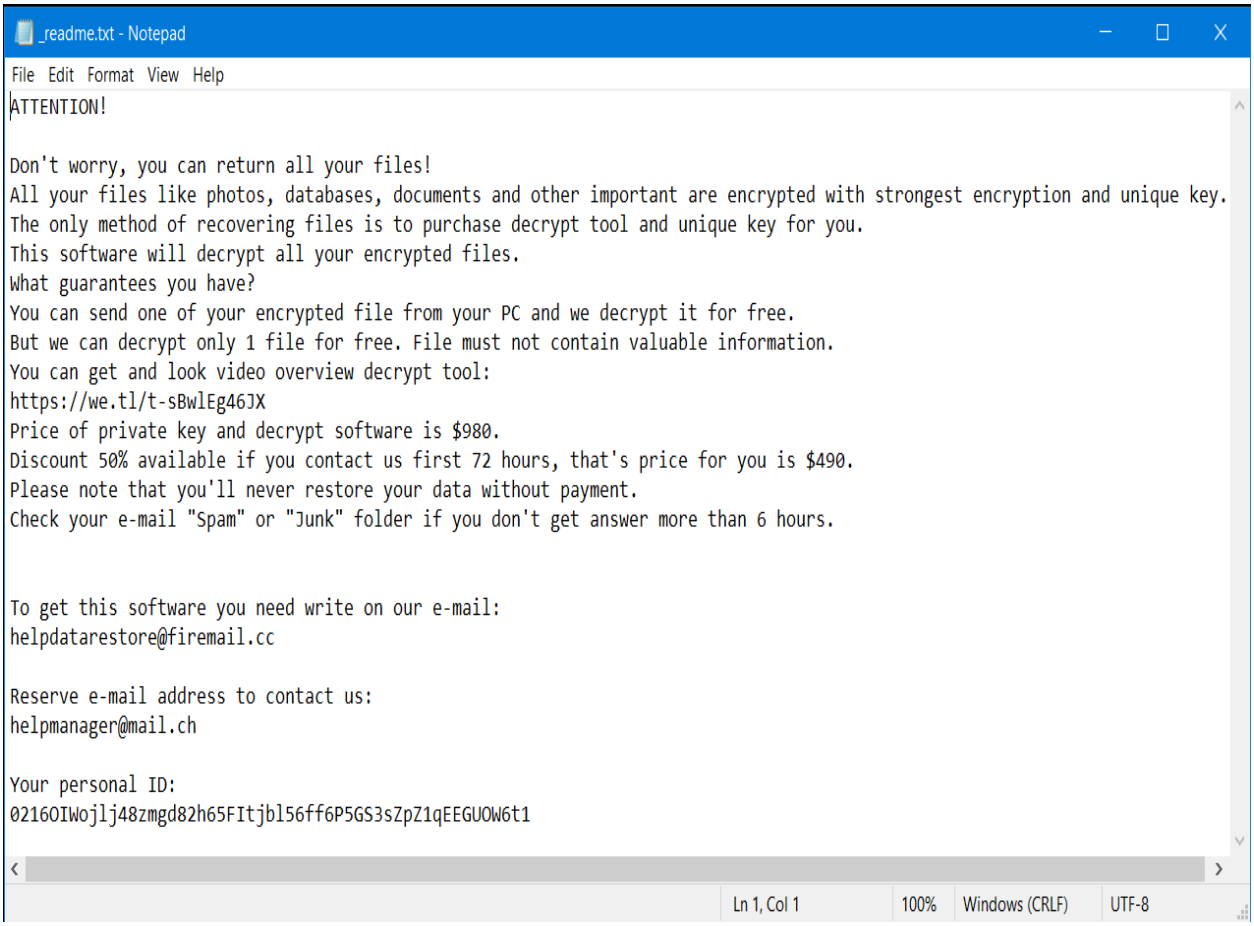
شناسه MD5	e93a3705fdd01cf942e4830e148f0b66	
شناسه SHA-1	4F640DA558A758DD5D58483C150E6D41A05F3766	
شناسه SHA-256	f6eb4d6bdb959f3ac087e153db2077a0372d3ed984f32184ff04d273ffdfc64c	
پلتفرم هدف	سیستم‌عامل ویندوز (WIndows-based OS)	
نوع بدافزار	باج‌افزار «Ransomware»	
اسکریپت شناسایی	یارا	ندارد
	اسنورت	ندارد
توضیحات بدافزار	این بدافزار که توسط آزمایشگاه امنیت کی‌پاد «باج‌افزار Speedy» نامگذاری شده است، که توسط ضد باج‌افزار کی‌پاد قابل شناسایی است.	

فهرست

مقدمه باج‌افزار Speedy

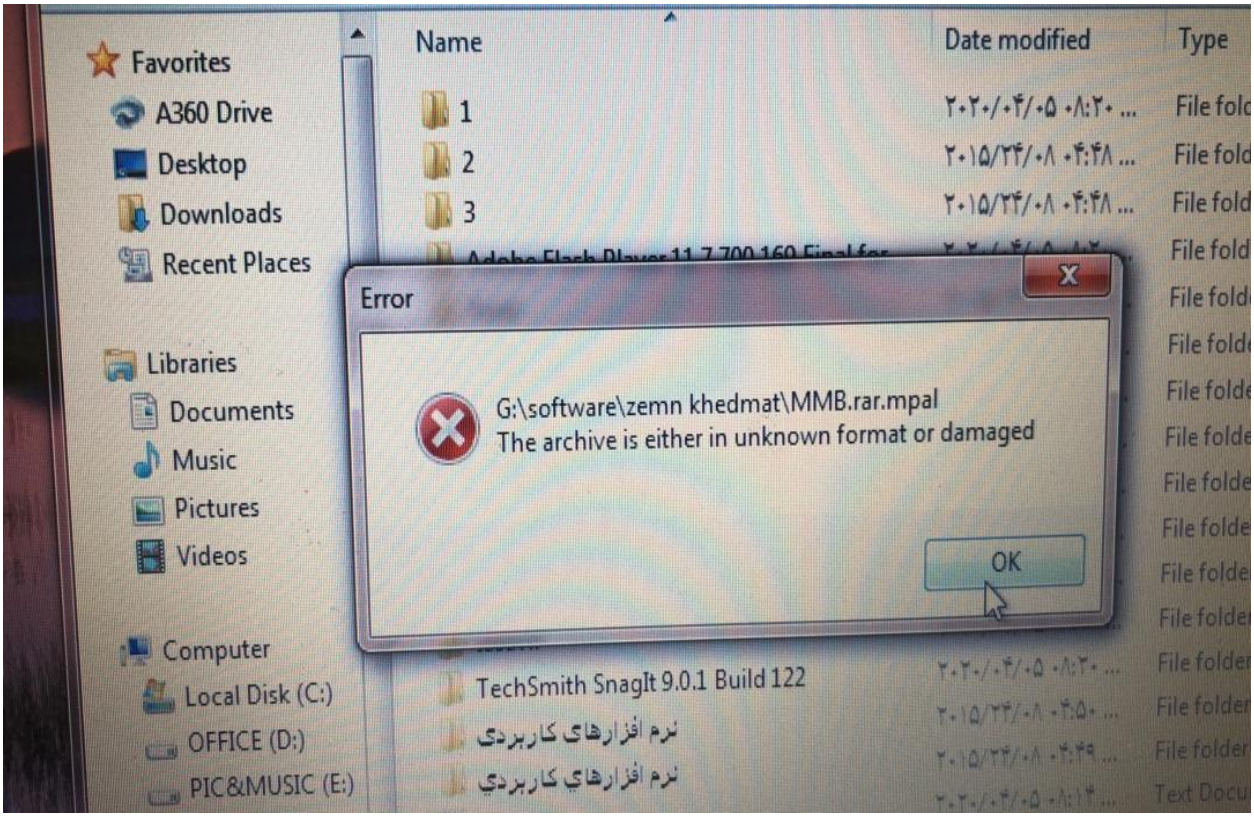
این نسخه باج‌افزار که با عنوان Speedy نامگذاری شده است، بعد از اینکه عملیات خود را آغاز می‌کند، تمامی فایل‌ها را با فرمت opqz رمزنگاری خواهد کرد. البته Speedy یکی از باج‌افزارهای عملیاتی از خانواده STOP/DJVU است. همچنین در هر پوشه یک فایل readme.txt_ ایجاد می‌کند که حاوی راهنما برای پرداخت باج و بازیابی فایل‌های رمزنگاری شده است.

با توجه به متن موجود در این فایل راهنما، مبلغ باج درخواستی ۹۸۰ دلار است، اما اگر در ۷۲ ساعت ابتدایی مبلغ باج را پرداخت کنید، ۵۰ درصد تخفیف به شما تعلق خواهد گرفت. با این حال، تاکنون گزارشی از دریافت کلید رمزگشایی فایل‌ها بعد پرداخت باج وجود ندارد. همچنین هیچگاه توصیه نمی‌شود که مبلغ باج توسط قربانیان پرداخت شود چون به آن‌ها فرصت می‌دهد که مجدد شما را مورد حمله قرار بدهند.



تصویر ۱: متن باج‌خواهی باج‌افزار Speedy

البته نمونه دیگری از فعالیت‌های مشابه باج‌افزارهای خانواده STOP/DJVU در موقعیت‌های دیگر هم مشاهده شده است، به عنوان مثال در تصویر ۲ وضعیت کامپیوتر یک قربانی را مشاهده می‌کنید که توسط باج‌افزاری از خانواده STOP/DJVU مورد هدف قرار گرفته است و تمامی فایل‌های مهم او با فرمت mpal. رمزنگاری شده است.



تصویر ۲: آلودگی ماشین یک کاربر به باج‌افزار

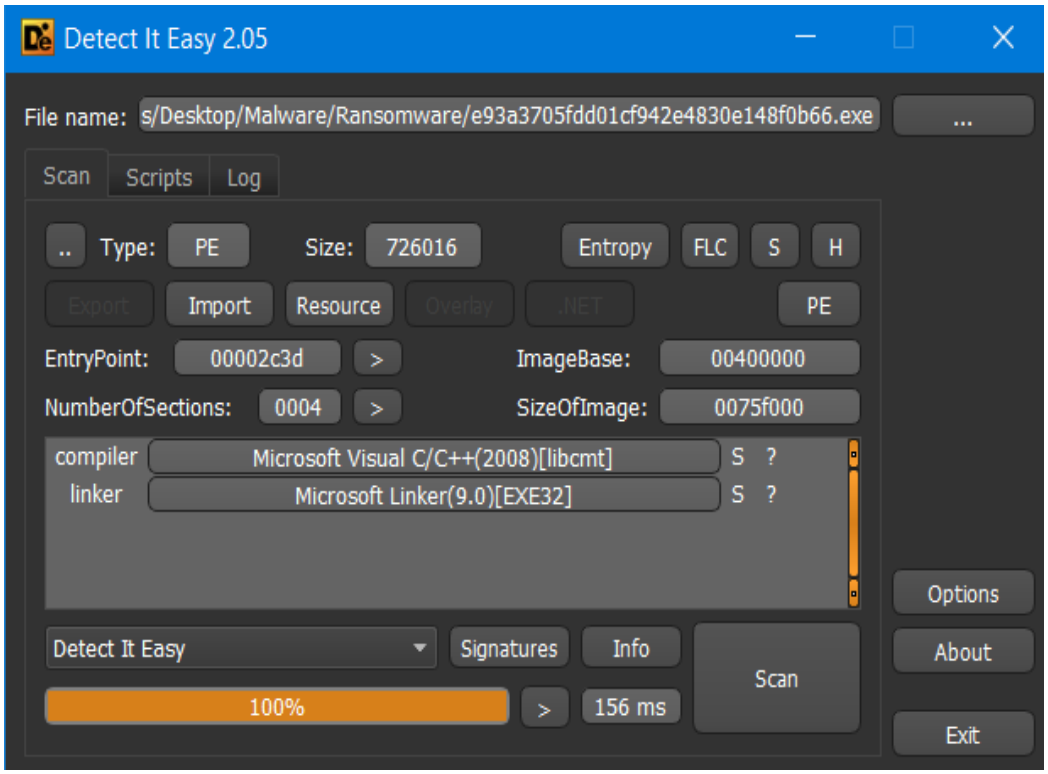
به هر صورت، از این خانواده بدافزارهای مشابه با نام هایی از قبیل STOP، Puma، DJVU، Drume و StopData تاکنون مشاهده شده است که تقریبا یک ساختار و عملکرد مشابه با یکدیگر دارند و فرمت فایل opqz، mpal، mogranos، cosakos، nvetud، lotej، kovasoh، prandel، zatrov، masok، brusaf، londec، یا krusop تنها بخشی از فرمت فایل‌هایی هستند که این بدافزار استفاده می‌کند.

تجزیه و تحلیل بدافزار Speedy

همانطور که در قسمت بالا ذکر شد، خانواده باج‌افزارهای STOP/DJVU با نام‌هایی از قبیل STOP و DJVU و ... تاکنون مشاهده شده‌اند که تقریبا یک ساختار و عملکرد مشابه با یکدیگر دارند و فرمت فایل opqz یا mpal فرمت فایل‌هایی هستند که این بدافزار حداقل برای هدف قرار دادن سازمان‌ها و کاربران ایرانی تاکنون استفاده کرده است.

اطلاعات فایل اجرایی بدافزار

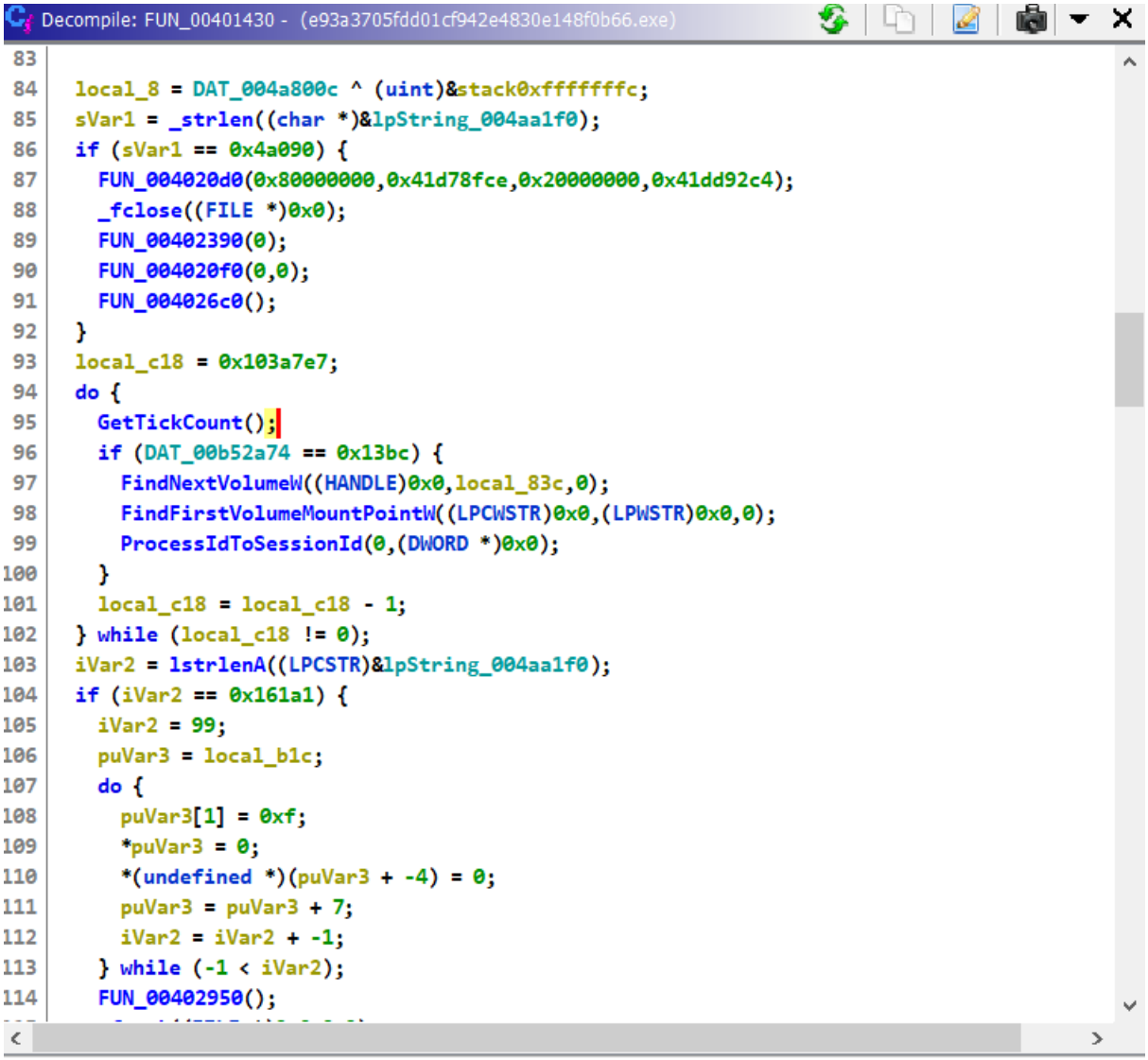
نمونه باجافزاری که از این خانواده قرار است، مورد بررسی و تحلیل قرار بدهیم، مانند نمونه‌های پیشین با زبان CPP و کامپایلر MSVC نوشته و ایجاد شده است که در تصویر ۳ این اطلاعات را می‌توانید مشاهده کنید. شایان ذکر است، شناسه MD5 این بدافزار پیش از این در جدول معرفی آن ارائه شده است که می‌توانید از آن برای دنبال کردن فعالیت‌های این بدافزار یا مشاهده دیگر گزارش‌های مرتبط با این بدافزار استفاده کنید.



تصویر ۳: اطلاعات فایل PE بدافزار

هنگامیکه دراپر این بدافزار اجرا می‌شود، در گام اول تابع GetThickCount را فراخوانی می‌کند که کاربرد آن با محوریت ضددیباگ است. به عبارت دیگر، توسعه‌دهندگان بدافزار از آن استفاده می‌کنند، تا اجازه تحلیل بدافزار را به تحلیل‌گران و یا سندباکس‌ها ندهند. از همین روی، وقتی فایل اجرایی دراپر باجافزار را تحلیل و بررسی می‌کنیم، در گام اول فراخوانی این تابع را مشاهده خواهیم کرد.

در تصویر ۴، فراخوانی این تابع نمایش داده شده است. بعد از اینکه این تابع فراخوانی شود، مقدار زمانی که بازگشت می‌دهد، مدت زمانی است که ماشین عملیاتی بوده است. در نتیجه بدافزار به سادگی با بررسی این عدد بازگشتی می‌تواند تشخیص بدهد که بر روی چه نوع سامانه‌ای قرار گرفته است. اگر این مدت زمان کم باشد، فعالیت خود را ملقی می‌کند، چون احتمال وجود سندباکس در این شرایط وجود دارد.



تصویر ٤: فراخوانی تابع GetThickCount

بعد از اینکه تابع مذکور فراخوانی شود، روتینی اجرا خواهد شد که باینری دیگری را به درون حافظه هیپ تزریق خواهد کرد و در نهایت با استفاده از تکنیک Module Stomping آن را در حافظه به صورت On the fly اجرا خواهد کرد.

تکنیک Module Stomping

یکی از تکنیک‌هایی که در عموم باج‌افزارهای این خانواده مشاهده می‌شود، استفاده از تکنیک Module Stomping است که به مهاجم اجازه می‌دهد کدی (مثلا یک فایل باینری) را در فضای هیپ تزریق و در نهایت اجرا کند، که در حال عادی این کار ممکن نیست چون فضای هیپ فقط خواندنی (Read-Only و Writeable) است.

در این نسخه از باجافزار با مشاهده فرخوانی توابع سیستمی ویندوز از قبیل VirtualProtect و VirtualAlloc می‌توان حدس زد که توسعه‌دهندگان این بدافزار مجدد از این تکنیک استفاده کرده‌اند تا کدی را به صورت On the fly بر روی حافظه اجرا کنند. در تصویر ۵، استفاده از این توابع توسط باجافزار مذکور نمایش داده شده است.

شایان ذکر است، توابع VirtualAlloc، VirtualProtect و VirtualFree در این لیست برای ما اهمیت فراوانی دارند، زیرا می‌توانیم با تنظیم یک Breakpoint بر روی آن‌ها تشخیص بدهیم که وقتی بدافزار از این رابط‌های برنامه‌نویسی ویندوز را فراخوانی می‌کند، چه هدفی در پس‌زمینه وجود دارد.

IDA View-A | Hex View-1 | Structures | Enums | Imports

Address	Ordinal	Name	Library
004A30F4		SetFilePointer	KERNEL32
004A30E4		SetHandleCount	KERNEL32
004A3134		SetLastError	KERNEL32
004A3168		SetStdHandle	KERNEL32
004A30CC		SetUnhandledExceptionFilter	KERNEL32
004A30AC		SetupComm	KERNEL32
004A3040		SizeofResource	KERNEL32
004A3100		Sleep	KERNEL32
004A30C0		TerminateProcess	KERNEL32
004A3124		TlsAlloc	KERNEL32
004A312C		TlsFree	KERNEL32
004A3120		TlsGetValue	KERNEL32
004A3128		TlsSetValue	KERNEL32
004A30C8		UnhandledExceptionFilter	KERNEL32
004A3164		VirtualAlloc	KERNEL32
004A3144		VirtualFree	KERNEL32
004A3098		VirtualProtect	KERNEL32
004A3090		WTSGetActiveConsoleSessionId	KERNEL32
004A303C		WideCharToMultiByte	KERNEL32
004A319C		WriteConsoleA	KERNEL32
004A31A4		WriteConsoleW	KERNEL32
004A310C		WriteFile	KERNEL32
004A3014		_lwrite	KERNEL32
004A3054		lstrcatA	KERNEL32
004A3060		lstrlenA	KERNEL32

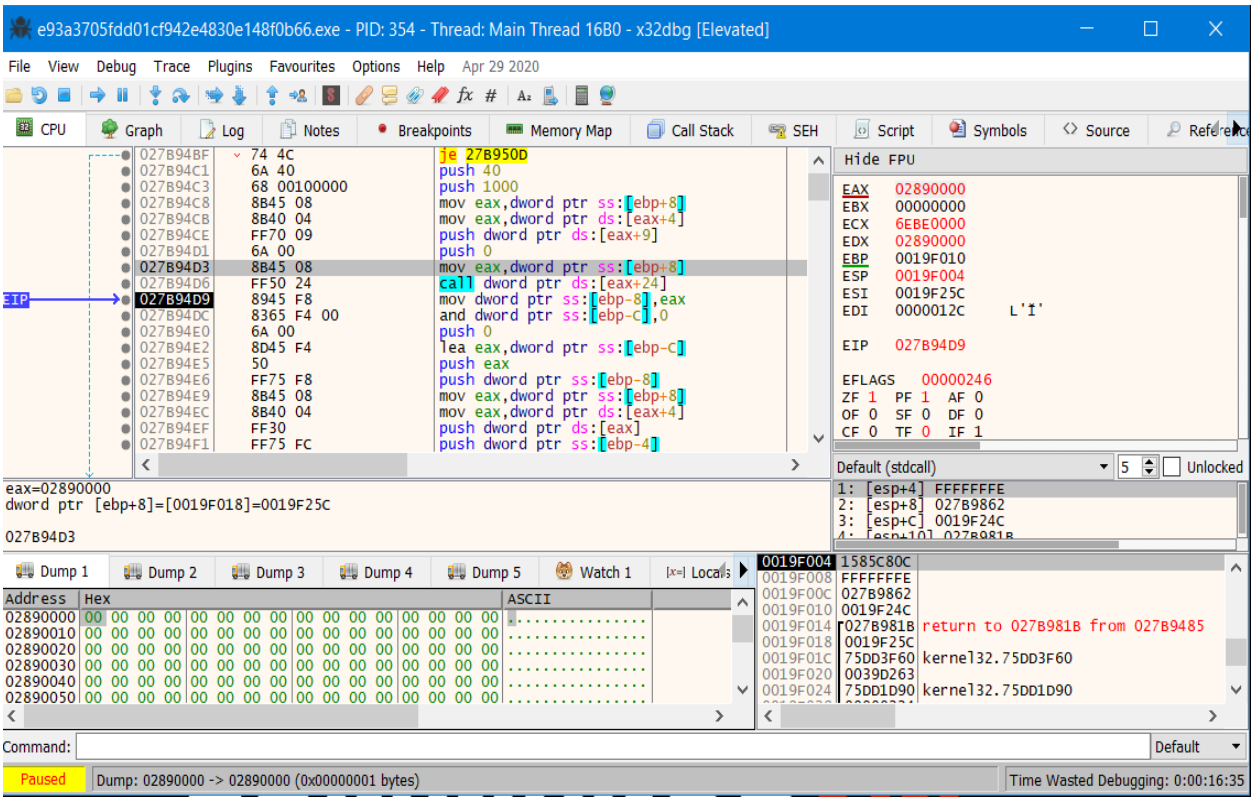
تصویر ۵: توابع وارد شده به فایل اجرایی

با بررسی این توابع می‌توانیم سوال‌های گوناگونی را پاسخ بدهیم از قبیل اینکه آیا قرار است کدی توسط بدافزار به درون حافظه هیپ تزریق و اجرا شود؟ اگر چنین است، آن کد چه کاری انجام می‌دهد و چه ساختاری دارد؟ با تحلیل و به دست

آوردن این دست اطلاعات بهتر می‌توانیم با ساختار داخلی این بدافزار آشنا شویم و در نهایت برای آن یک راه‌حل پایدار دفاعی ارائه کنیم.

در تصویر ۶، فرخوانی تابع VirtualAlloc را مشاهده می‌کنید که بعد از فراخوانی آن قسمتی از حافظه هیپ به بدافزار تخصیص داده خواهد شد که می‌توان در آن کدی را تزریق کرد و در نهایت با فراخوانی تابع VirtualProtect به آن سطح دسترسی Executable داد.

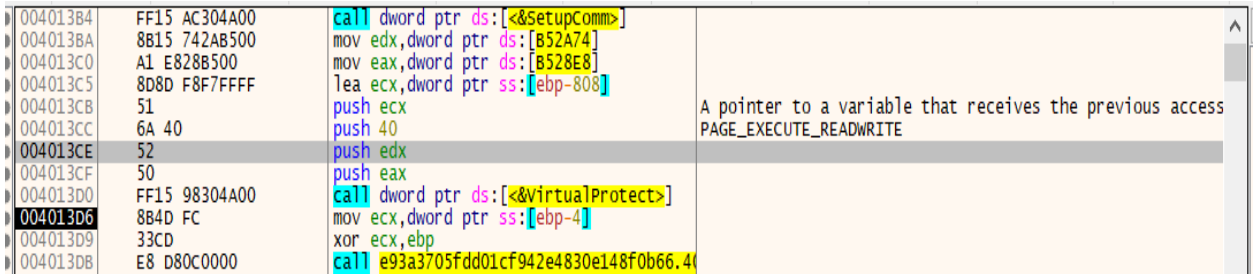
پس از اینکه با موفقیت سطح دسترسی Executable به آن صفحه از حافظه (Memory Page) ارائه شود، بدافزار می‌تواند کد قرار گرفته در آن بلاک از حافظه را به صورت On the Fly بر روی حافظه اجرا کند. در تصویر ۵، فرخوانی VirtualAlloc و ناحیه‌ای از حافظه هیپ که بعد فراخوانی این رابط برنامه‌نویسی به بدافزار تخصیص داده شده است، مشاهده می‌کنید.



تصویر ٦: فراخوانی VirtualAlloc و حافظه تخصیص از آن

بعد از فراخوانی VirtualAlloc، حافظه هیپ درخواستی به بدافزار تخصیص داده خواهد شد. بعد از فراخوانی این رابط برنامه‌نویسی، آدرسی که در رجیستر EAX ذخیره می‌شود، آغاز حافظه تخصیص داده شده به بدافزار را نمایش می‌دهد. شایان ذکر است، اگر به این آدرس در محیط Dump رجوع کنید، محتویات این حافظه تخصیص داده شده را مشاهده خواهید کرد که با مقادیر NULL یا 00 پر شده‌اند.

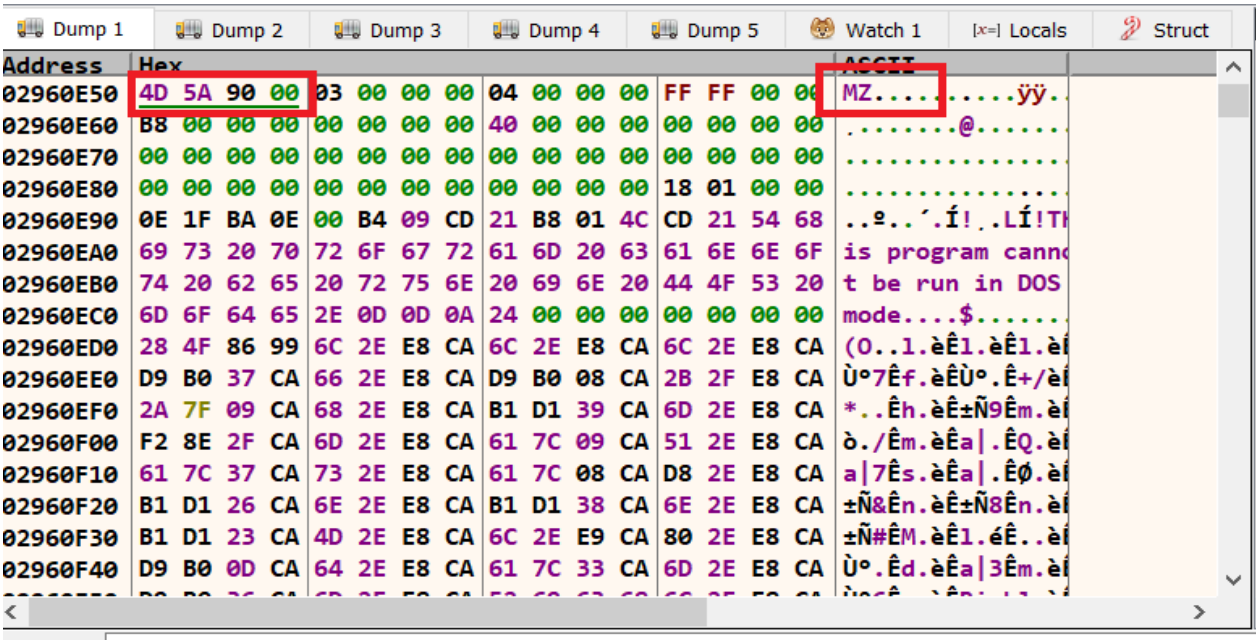
در گام بعد وقتی تکه کد مورد نظر توسعه‌دهندگان بدافزار در این قسمت از حافظه قرار گرفت، تابع VirtualProtect را فراخوانی خواهند کرد تا به آن ناحیه از حافظه هیپ سطح دسترسی PAGE_EXECUTE_READWRITE بدهند که در نهایت بتوان در آن ناحیه کدی را اجرا کرد (در اینجا یک باینری است که عمل رمزنگاری فایل‌ها را بر عهده دارد). در تصویر ۷، پارامترهایی عبوری به VirtualProtect نمایش داده شده است.



تصویر ۷: پارامترهای عبوری به رابط VirtualProtect

بعد از اینکه VirtualAlloc و VirtualProtect با موفقیت کار خود را انجام بدهند، در حافظه هیپ تخصیص داده شده توسط VirtualAlloc، کدی را که توسط بدافزار در ادامه اجرا خواهد شد، خواهیم دید. در ادامه می‌توانیم با دامپ گرفتن از آن کد به صورت مستقل در قالب یک فایل باینری مجزای دیگر تحلیل آن را آغاز کنیم.

همانطور که در تصویر ۸ در قسمت پنجره Dump مشاهده می‌کنید، کدی که به درون حافظه هیپ تزریق شده است و توسط VirtualProtect به آن دسترسی Executable ارائه شد، در شروع خود دارای فرمت MZ است که به عنوان امضای فایل‌های PE در ویندوز شناخته می‌شود.

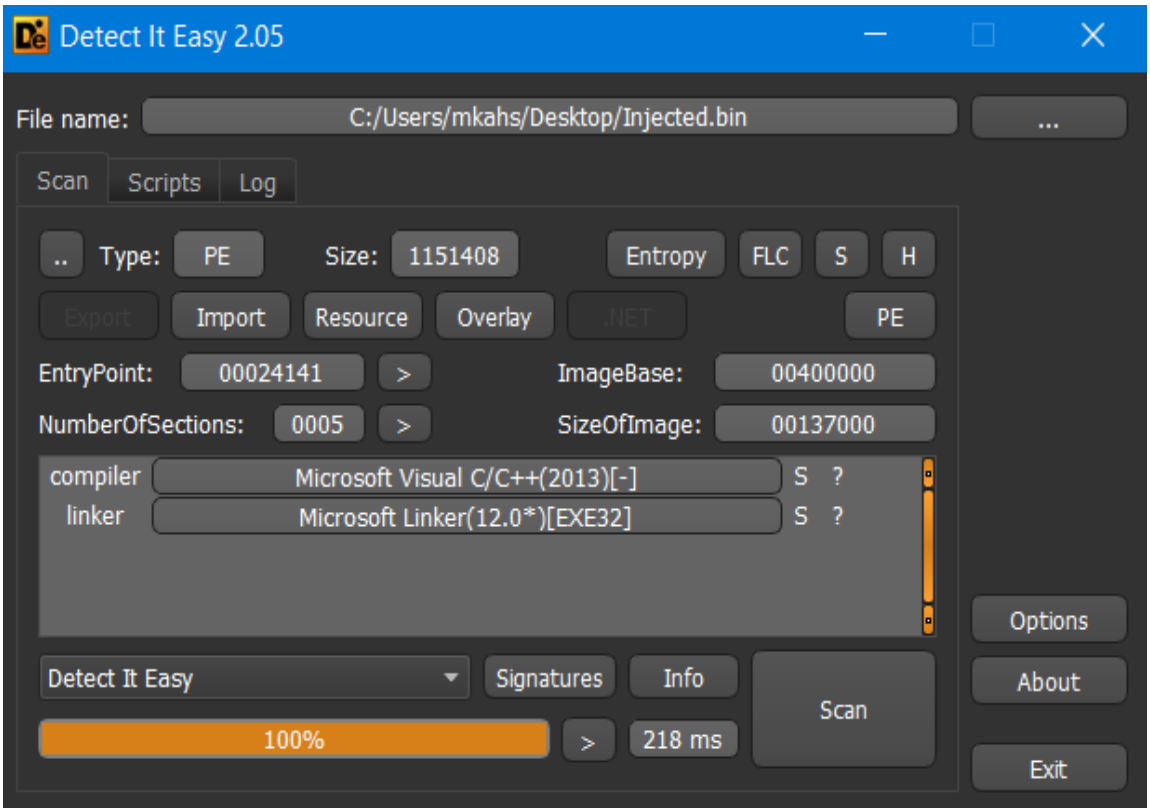


تصویر ۸: کد تزریق شده به حافظه هیپ توسط بدافزار

تحلیل کد تزریق شده به حافظه هیپ

همانطور که پیش از این ذکر شد، برای ادامه تجزیه و تحلیل باجافزار، نیازمند هستیم از فایل تزریق شده به درون حافظه هیپ توسط بدافزار دامپ بگیریم. در این قسمت، آن را با عنوان Injected.bin، در قالب یک فایل اجرایی مجزا از روی حافظه دامپ گرفتیم تا سپس به تجزیه و تحلیل آن بپردازیم.

همانطور که در تصویر ۹ اطلاعات این فایل اجرایی نمایش داده شده است، مجدد با زبان CPP و کامپایلر MSVC برنامه‌نویسی و کامپایل شده است. همچنین با توجه به آنتروپی آن به نظر می‌رسد که مبهم‌سازی هم نشده باشد که این موضوع به تنهایی کار تحلیل ساختار این فایل اجرایی را برای ما ساده‌تر می‌کند.



تصویر ۹: اطلاعات فایل تزریق شده به درون هیپ توسط باجافزار

حال که به کد تزریق شده به درون هیپ دسترسی مجزا داریم، می‌توانیم ساختار داخلی آن را به سادگی توسط ابزارهای گوناگون مشاهده کنیم. به عنوان مثال، اگر رشته‌های درون فایل اجرایی Injected.bin را مشاهده کنیم، اطلاعات جالبی را به دست خواهیم آورد از قبیل اینکه در این فایل اجرایی از زیرسیستم رمزنگاری Microsoft Enhanced RSA، AES Cryptographic Provider، کتابخانه OpenSSL و ثابت‌های انکدینگ و دکدینگ Netscape و ... به

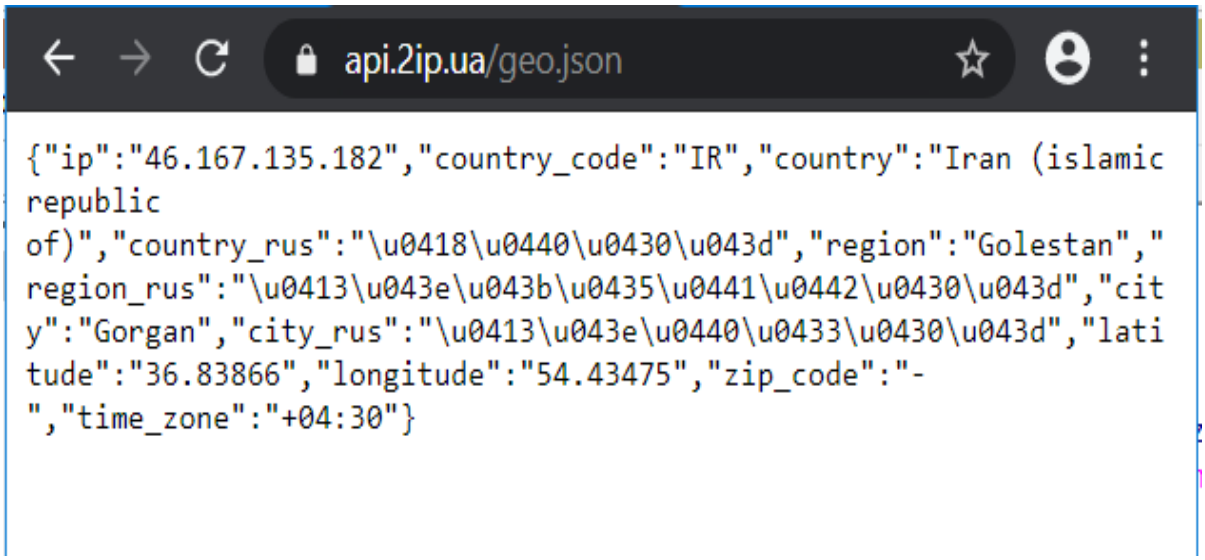
صورت گسترده استفاده شده است. در تصویر ۱۰، رشته هایی که گواه استفاده از MER، ACP و OpenSSL در فایل اجرایی هستند، نمایش داده شده است.

file	-	-	MoveFile
-	-	-	Montgomery Multiplication for x86, CRYPTOGAMS by <appro@openssl.org>
-	-	-	Monday
-	-	-	Monday
-	-	-	Modulus:
execution	Process Discovery	Discovery	Module32Next
execution	Process Discovery	Discovery	Module32First
-	-	-	MinghuaQuw
-	-	-	MinghuaQuS
-	-	-	MinghuaQu)
-	-	-	MinghuaQu
-	-	-	MinghuaQu
-	-	-	MinghuaQu
-	-	-	MinghuaQu
-	-	-	MinghuaQu
-	-	-	Microsoft Visual C++ Runtime Library
-	-	-	Microsoft Universal Principal Name
-	-	-	Microsoft Trust List Signing
-	-	-	Microsoft Smartcardlogin
-	-	-	Microsoft Server Gated Crypto
-	-	-	Microsoft Local Key set
-	-	-	Microsoft Internet Explorer
-	-	-	Microsoft Internet Explorer
-	-	-	Microsoft Individual Code Signing
-	-	-	Microsoft Extension Request
cryptography	-	-	Microsoft Enhanced RSA and AES Cryptographic Provider
-	-	-	Microsoft Encrypted File System
-	-	-	Microsoft Commercial Code Signing
-	-	-	Microsoft CSP Name

تصویر ۱۰: رشته‌های متعلق به فایل باینری Injected.bin

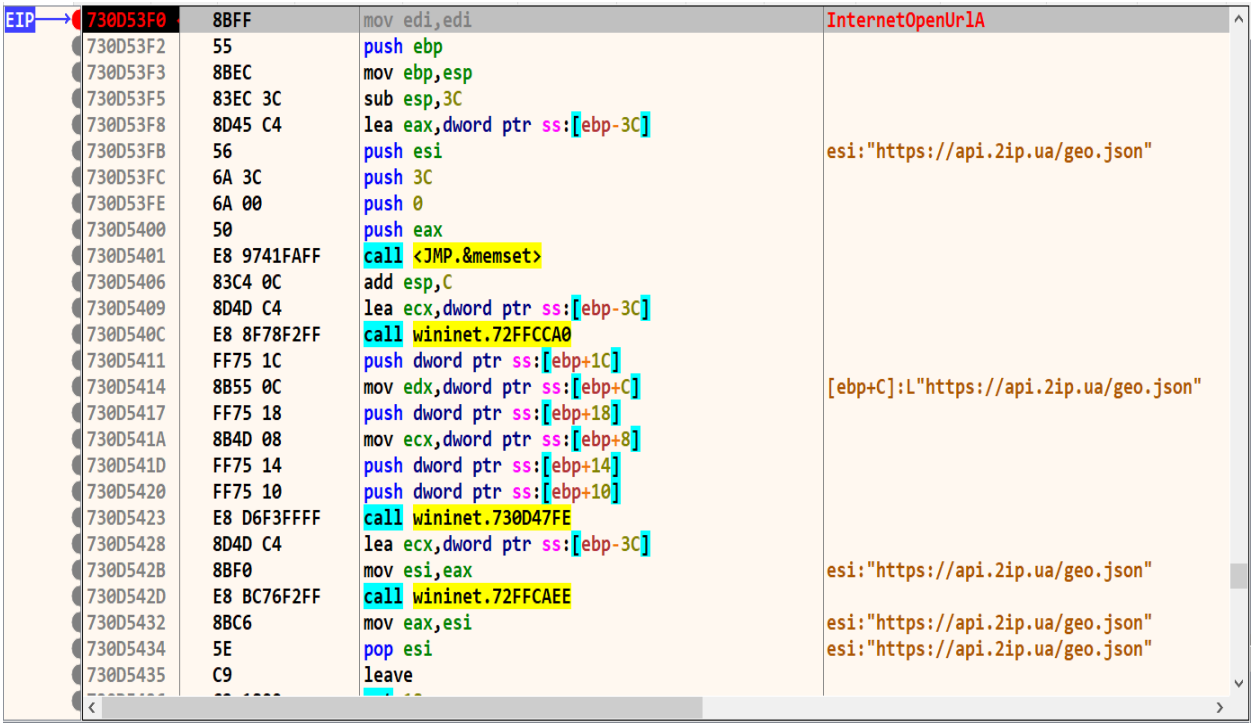
در ادامه، اگر فایل اجرایی Injected.bin را دیزاسمبل کنیم و ساختار آن را مورد بررسی قرار بدهیم، مشاهده خواهیم کرد که در گام اول اجرای خود به متغیرهای محلی سیستم دسترسی می‌گیرد و مقادیری که درون بلاک محیط پروسه (Process Environment Block) وجود دارد، بررسی و پردازش می‌کند.

در مرحله بعدی، وارد روتینی خواهد شد که با وبگاه https://api.2ip.ua/geo.json ارتباط خواهد گرفت، تا اطلاعات ماشین قربانی از قبیل آدرس IP، کد کشور، نام کشور، نام شهر، منطقه، مختصات جغرافیایی و ... را استخراج کند. در تصویر ۱۱، اطلاعاتی بازگشتی توسط این وب‌سایت هنگام برقراری ارتباط با آن نمایش داده شده است.



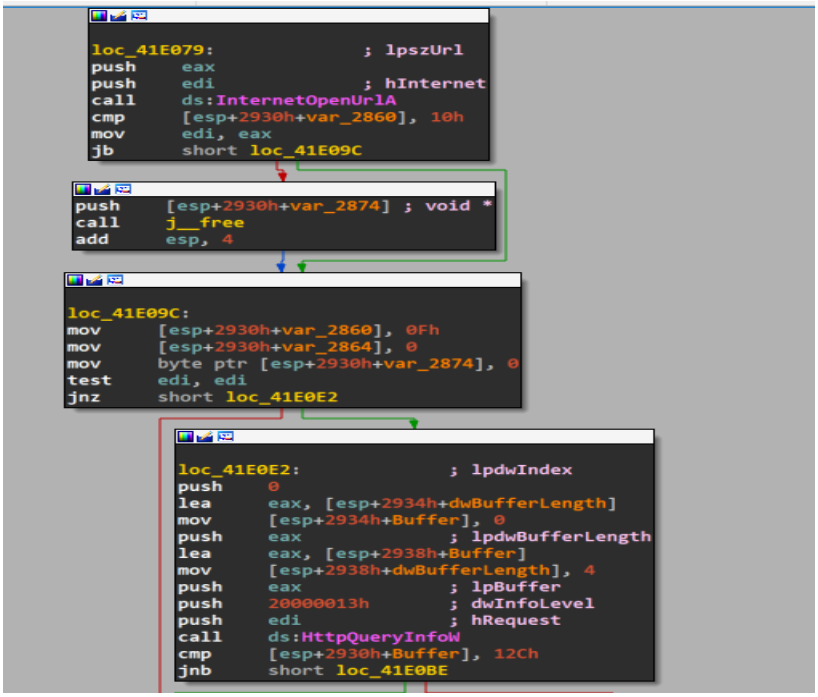
تصویر ۱۱: اطلاعات بازگشتی توسط سایت 2ip.ua

در تصویر ۱۲، قسمتی از کد فایل اجرایی Injected.bin را در محیط دیباگر مشاهده می‌کنید که بدافزار با استفاده از رابط برنامه‌نویسی InternetOpenUrl تلاش به برقراری ارتباط با وب‌سایت 2ip.ua و خواندن اطلاعات بازگشتی توسط این وبسایت در قالب یک فایل JSON کرده است.



تصویر ۱۲: برقراری ارتباط با 2ip

در تصویر ۱۳، فراخوانی تابع HttpQueryInfoW را مشاهده می‌کنیم که بدافزار با استفاده از این رابط برنامه‌نویسی اطلاعات بازگشتی توسط وب‌سایت 2ip.ua را می‌خواند و پردازش می‌کند. وقتی وب‌سایت 2ip.ua اطلاعات ماشین کاربر را بازگشت داد، بدافزار با بررسی کد کشور قربانی کار خود را ادامه می‌دهد.



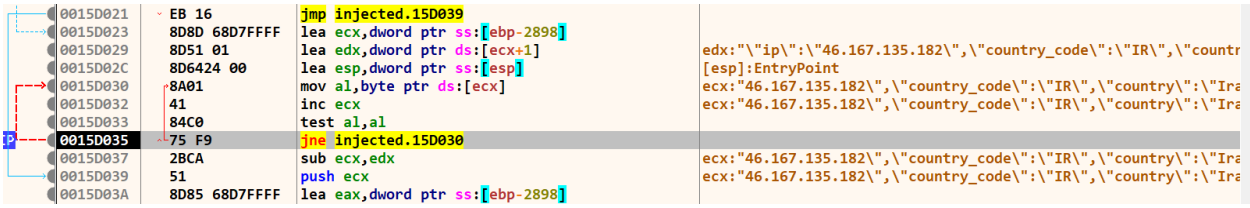
تصویر ۱۳: استفاده از HttpQueryInfoW

در تصویر ۱۴، این مسئله نمایش داده شده است. همینطور که در این تصویر مشاهده می کنید، بعد اینکه اطلاعات بازگشتی توسط بدافزار خوانده می شود، باج افزار به تابع 1656D0 وارد خواهد شد تا این اطلاعات را بررسی دقیق کند.

```
0015CFB4   E9 5A020000     jmp injected.15D213
0015CFB9   8D85 68FFFFFF   lea eax,dword ptr ss:[ebp-98]
0015CFBF   50              push eax
0015CFC0   68 00280000     push 2800
0015CFC5   8D85 68D7FFFF   lea eax,dword ptr ss:[ebp-2898]
0015CFCB   50              push eax
0015CFCC   56              push esi
0015CFCD   FF15 A8C32100   call dword ptr ds:[<&InternetReadFile>]
0015CFD3   56              push esi
0015CFD4   8B35 A4C32100   mov esi,dword ptr ds:[<&InternetCloseHandle>]
0015CFDA   FFD6            call esi
0015CFDC   57              push edi
0015CFDD   FFD6            call esi
0015CFDF   6A 10           push 10
0015CFE1   68 ECFF2400     push injected.24FFEC                24FFEC:"\"country_code\":\""
```

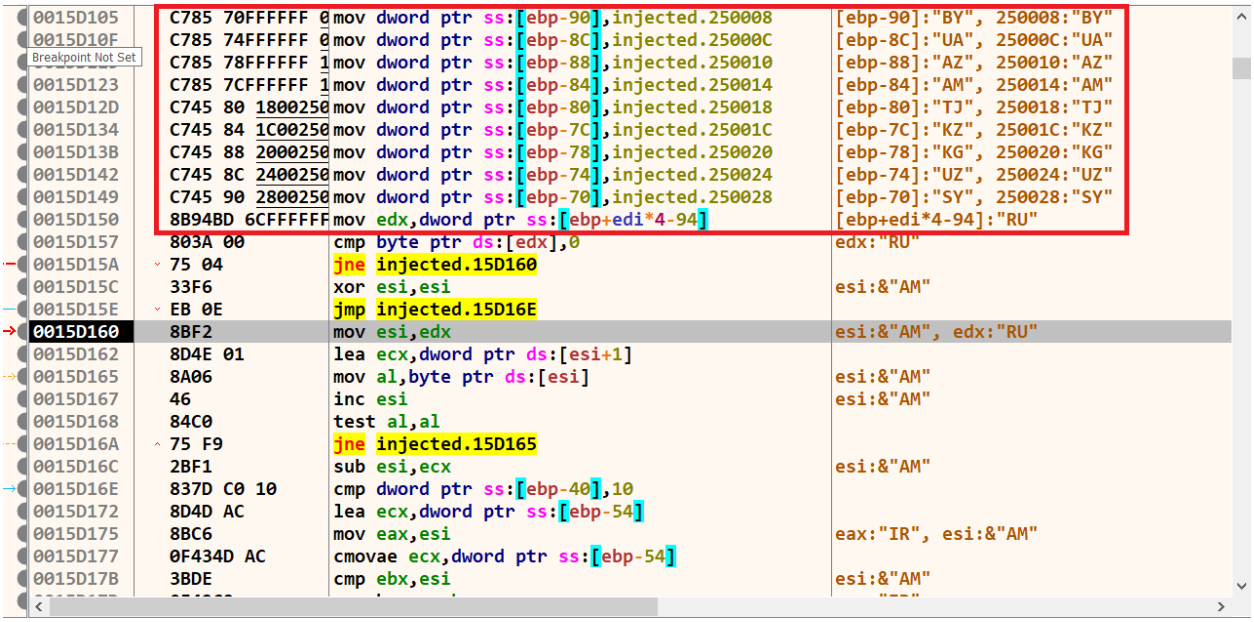
تصویر ۱۴: عبور کد کشور قربانی به یک روتین دیگر

در تصویر ۱۵، نحوه پردازش اطلاعات را مشاهده می کنید. همانطور که در این تصور نمایش داده شده است، اطلاعات بازگشتی توسط وب سایت 2ip.ua توسط بدافزار بایت به بایت مورد پردازش قرار می گیرد تا در ادامه بتواند با استفاده از این اطلاعات به شیوه دقیق تری با قربانی تعامل برقرار و از آن باج خواهی کند.



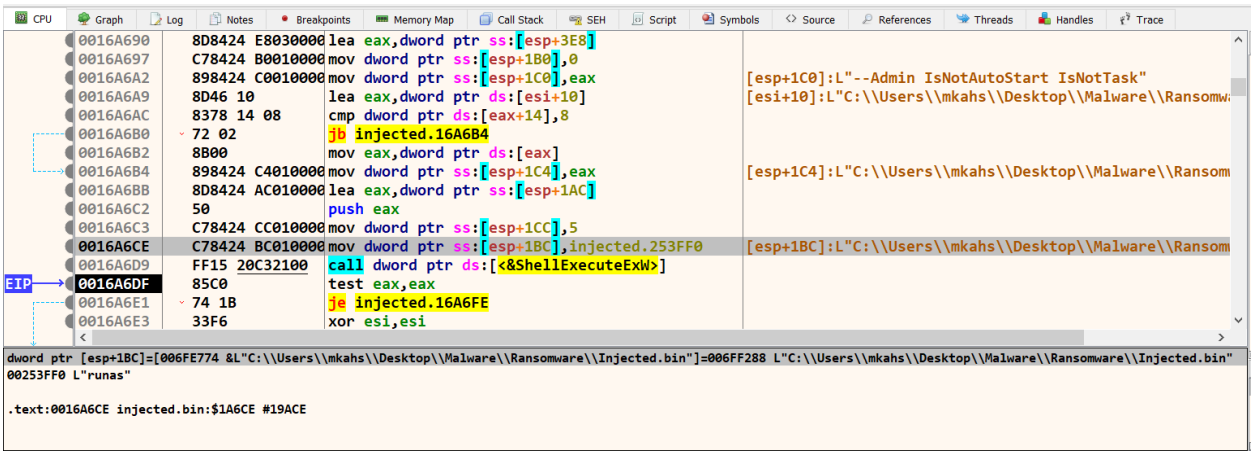
تصویر ۱۵: پردازش اطلاعات بازگشتی توسط 2ip.ua

در گام بعد کد کشور قربانی با کدهای کشوری که در بدافزار نهفته شده است، مقایسه خواهد شد تا اگر قربانی دارای کد کشوری مشابه با لیست نهفته شده در بدافزار بود، نوع عملکرد باج افزار شخصی سازی و هدفمندتر شود. در تصویر زیر، کادر قرمز رنگ، کد کشورهایی است که در بدافزار نهفته شده اند و مقدار IR کد کشور قربانی (ماشین تحلیلگر بدافزار) است.



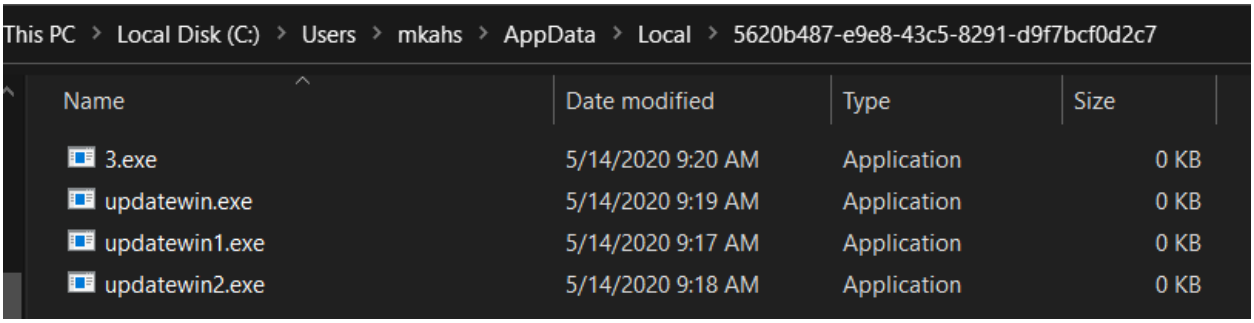
تصویر ۱۶: کد کشورهای نهفته شده در باج افزار

در گام بعد، فایل اجرایی Injected.bin برای اینکه با موفقیت بتواند عملیات های مورد نظر خود را انجام بدهد، رابط برنامه نویسی ShellExecute را به همراه پارامتر runas اجرا خواهد کرد تا دسترسی خود را افزایش دهد. در تعریف این رابط برنامه نویسی در مایکروسافت آورده شده است که این رابط برنامه نویسی هنگامیکه با پارامتر runas اجرا می شود، یک برنامه کاربردی را با سطح دسترسی Administrator اجرا خواهد کرد. در تصویر ۱۷، فرخوانی ShellExecute توسط بدافزار نمایش داده شده است.



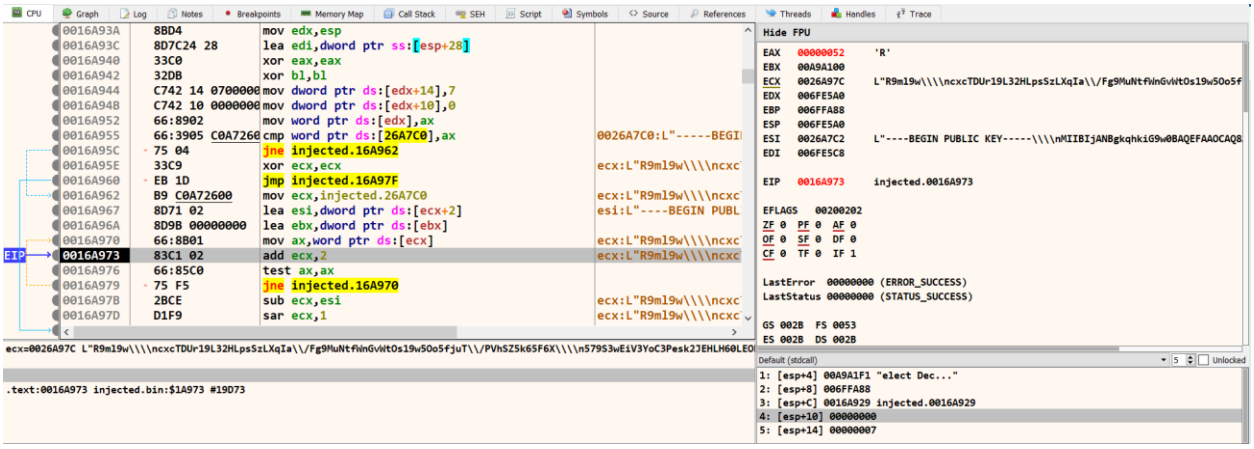
تصویر ۱۷: اجرای پیلود اجرایی بدافزار توسط رابط ShellExecute

سپس بدافزار تلاش خواهد کرد از یک وب سایت با آدرس www.nokd.top چند فایل اجرایی دانلود و در مسیر AppData ذخیره سازی کند. این فایل ها با نام های updatewin.exe، updatewin1.exe، updatewin2.exe، 3.exe و 5.exe هستند که به دلیل غیرفعال بودن وب‌سایت nokd.top این عملیات با موفقیت همراه نخواهد شد و تمامی فایل هایی که بر روی مسیر AppData ایجاد می شوند، خراب خواهند بود. در تصویر ۱۸، فایل هایی که توسط بدافزار بر روی مسیر AppData قرار گرفته اند، نمایش داده شده است.



تصویر ۱۸: فایل‌های مرتبط با بدافزار Injected.bin

سپس برای شروع رمزنگاری کلید عمومی (Public Key) را ایجاد می کند و سپس بایت به بایت آن را مورد بررسی قرار می دهد. در تصویر ۱۹ مقدار کلید عمومی را مشاهده می کنید که در یک حلقه بایت به بایت آن مورد بررسی قرار می گیرد.



تصویر ۱۹: کلید عمومی ایجاد شده توسط بدافزار

کلید عمومی که توسط بدافزار ایجاد شده است، در زیرآورده شده است. بعد ایجاد این کلید عمومی، در هر دیرکتوری که بدافزار عمل رمزنگاری انجام می دهد، یک فایل با نام _readme.txt ایجاد خواهد شد که در برگیرنده اطلاعات تماس به توسعه دهندگان بدافزار و شناسه ماشین قربانی و ... است.

-----BEGIN PUBLIC KEY-----

\\\\nMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA6mc4av0qBgVc00vu
120E\\\\n6TPCO04Qbf79EbDUdCP3FO5jFuLLmOa15oplHveKsXedyzhurGAunV6wJ
hjWxGxK\\\\nD43w210FT8uH\\/zZPqDvnatctgugSZHxRLMhBldi6JMz6hrYVyRDXL1
y7juR9ml9w\\\\ncxcTDUr19L32HLpsSzLXqIa\\/Fg9MuNtfWnGvWtOs19w5Oo5fju
T\\/PVhSZ5k65F6X\\\\n579S3wEiV3YoC3Pesk2JEHLH60LEOPqlWaHbvlDGODi3U2
hDIYNENZFSFyS5dqaX\\\\nFEFdOpEqya8Nh77OEIK91fCWYXmK7I2UmZwPOtLihST
4iweNNMRmaw8p7blgLHbJ\\\\n5QIDAQAB\\\\n

-----END PUBLIC KEY-----

در نهایت وقتی بدافزار تمامی اطلاعات مربوط به فایل ها و دیرکتوری ها را به دست آورد، فرآیند رمزنگاری با استفاده از توابعی از قبیل CryptAcquireContext، CryptStringToBinary، CryptEncrypt و ... را شروع می‌کند. در تصویر ۲۰، روتینی که عمل رمزنگاری را انجام می دهد، آورده شده است.

در ابتدا CryptAcquireContext فرخوانی می‌شود تا یک هندل به CSP به دست آورد. شایان ذکر است، سرویس CSP یا ارائه دهنده سرویس رمزنگاری مایکروسافت (Cryptographic Service Provider) یک کتابخانه نرم‌افزاری است که رابطهای رمزنگاری مایکروسافت (Microsoft CryptoAPI) را پیاده‌سازی کرده است. این کتابخانه به توسعه‌دهندگان نرم‌افزار و بدافزار اجازه می‌دهد تا داده‌های خود را رمزنگاری و رمزگشایی کنند.

به هر صورت، همانطور که در تصویر ۲۰ قابل مشاهده است، بدافزار در گام اول یک آبجکت هش (Hash Object) از نوع MD5 ایجاد می‌کند. بعد هش MD5 مقدار درون متغیر V3 را محاسبه خواهد کرد (متغیر V4 همچنین طول متغیر V3 است). بعد مقدار هش ایجاد شده را با CryptGetHashParam می‌خواند و در متغیر v5 قرار می‌دهد.

```
26  v15 = 0;
27  sub_4156D0(a3, &unk_4FFCA4, 0);
28  v20 = 0;
29  if ( !CryptAcquireContextW(&phProv, 0, 0, 1u, CRYPT_VERIFYCONTEXT) )
30  {
31    v14 = 0;
32    _CxxThrowException(&v14, &_TI1H);
33  }
34  if ( !CryptCreateHash(phProv, 0x8003u, 0, 0, &phHash) )
35  {
36    v13 = 0;
37    _CxxThrowException(&v13, &_TI1H);
38  }
39  if ( !CryptHashData(phHash, v3, v4, 0) )
40  {
41    v12 = 0;
42    _CxxThrowException(&v12, &_TI1H);
43  }
44  pdwDataLen = 0;
45  if ( !CryptGetHashParam(phHash, 2u, 0, &pdwDataLen, 0) )
46  {
47    v11 = 0;
48    _CxxThrowException(&v11, &_TI1H);
49  }
50  v5 = (unsigned __int8 *)operator new[](pdwDataLen + 1);
51  v15 = v5;
52  memset(v5, 0, pdwDataLen + 1);
53  if ( !CryptGetHashParam(phHash, 2u, v5, &pdwDataLen, 0) )
54  {
55    v10 = 0;
56    _CxxThrowException(&v10, &_TI1H);
57  }
58  for ( i = 0; i < pdwDataLen; ++i )
59  {
60    sprintf(&v9, "%.2X", v5[i]);
61    if ( v9 )
62      sub_413EA0(&v9, strlen(&v9));
63    else
64      sub_413EA0(&v9, 0);
65  }
66  j_j__free(v5);
67  CryptDestroyHash(phHash);
68  CryptReleaseContext(phProv, 0);
69  return 1;
```

تصویر ۲۰: روتین رمزنگاری استریم‌های داده‌ای باج‌افزار

در نهایت اگر مقدار هش که محاسبه و خوانده شده است، دارای خطا یا نامعتبر باشد، یک exception ایجاد می‌شود. در غیر این صورت، هر بایت هش به معادل رشته‌ای خود تبدیل می‌شود، و اگر معادل رشته‌ای معتبر بود (یعنی مخالف NULL بود)، تابع sub_413EA0 صدا زده خواهد شد و معادل رشته‌ای، به عنوان آرگومان به آن ارائه می‌شود. این روال برای تمام بایت‌های هش تکرار می‌شود. در آخر هم آبجکت هش پس از این که از حلقه for بیرون آمدیم، از روی حافظه پاک و آزاد خواهد شد.

نتیجه گیری

باج افزار Speedy که از خانواده STOP/DJVVU است، از هر نظر برای مبحث امنیت ارتباطات کشور جمهوری اسلامی ایران اهمیت راهبردی دارد، زیرا این باج‌افزار، توانسته است با گستردگی بسیار زیادی کاربران و سازمان های ایرانی را آلوده کند.

اگر چه به نظر می‌رسد، نسخه فعلی که از این باج افزار انتشار پیدا کرده است، آخرین نسخه از این باج افزار نباشد و در آینده نسخه‌های عملیاتی و خاص این بدافزار مجدد کاربران و زیرساخت فناوری اطلاعات ایران را هدف قرار بدهد. از همین روی، پیش از اینکه سامانه‌های زیرساختی کشور مورد حمله این بدافزار قرار بگیرند، باید گام‌هایی به منظور حفظ یکپارچگی و عملکرد آن‌ها برداشته شود.

نشانه نفوذگر «IOC»

Name:	Hash
Samples:	E93A3705FDD01CF942E4830E148F0B66 4F640DA558A758DD5D58483C150E6D41A05F3766 DCE16104C90479656ECEA5C63B5C4A3B ACFC8C08A67E915E0187713CC0912AEFF00560A5
URLs	http://nokd.top https://api.2ip.ua/geo.json http://www.openssl.org/support/faq.html

جدول ۳: نشان نفوذ «IOC»